1DV501 Introduction to programming
Final project report
Amelia Rossi
Nov 7, 2025

For the final project in this class, I wrote a text analyzer that reads and provides statistics about a given text. The grade I am striving for in this assignment is an A grade. I believe that I have completed all required tasks, as well as additional functionality, with relatively good code quality. I chose to work on the project alone because it gave me more flexibility with how I wrote the program, and what features I added. I spent about 30 hours on this project over the past couple of weeks.

My code is structured quite differently from what was taught in the class, as I have been writing Python for a few years and chose to code in my usual style. There are four important parts of the code: the main loop, file handling, statistics, TUI, and plotting. The main loop follows a simple pattern: select a book, then pick a kind of analysis or export a report. After viewing the analysis, you can choose to either export a plot, analyze a new book, or pick a new kind of analysis. The main function is longer than I would like, but I tried to keep the logic very straightforward while also providing reasonable options at every step. Files are handled with a pair of classes. The 'Book' class handles opening and reading the file as chunks, and the 'GutenbergHeader' class handles reading metadata from the headers of Gutenberg texts. The program works with plain text files, but because of the information provided in the headers of gutenberg ebooks, I could show information like titles and authors in the UI, which adds a nice level of polish.. I learned about and used Python's magic methods to implement the Book class as both a context manager and an iterator. This functionality allows the file to be opened and closed safely, as well as conveniently split and read as chunks. The statistics are read in approximately 8 KB chunks, always ending in a newline. These chunks are split up and analyzed to efficiently process the number of words, sentences, paragraphs, and other datapoints in each chunk. The 'Stats' class also contains methods to process the raw results from data structures such as dictionaries into useful data, like averages and top words. The TUI is the most unique and interesting part of the project. I chose to write a TUI because I like expressive terminal user interfaces, and I wanted to do something a bit different with the project. I started out trying to write my own TUI code with ANSI escape codes, but quickly ran into a variety of bugs, and chose to switch to curses from the Python standard library. Curses allowed me to easily add functionality with fewer bugs and edge cases, and use high-level structures like 'windows' instead of just printing text. I had seen and heard about ncurses before this project, but this was my first time actually using the library, and I really enjoyed it. The plotting was also totally new to me, as I had never used matplotlib before. I found the plotting to be surprisingly simple to use, and I was able to implement most of the required functionality based on a few examples from

the official docs. The project must be run from a terminal, so plots cannot be displayed in OS windows or in notebook cells. I had to figure out how to export images instead. In my opinion, this export behavior actually improves the user interface, as the user can make multiple plots for different books and compare them easily. If I had put more time into the project, I would have focused on code quality. Because I was writing this relatively quickly, I omitted some things I find necessary in high-quality code. Notably, I wish I had added unit tests to increase confidence about the correctness of the program's statistics. I would also like to refactor some parts of the codebase that are messy, in particular the TUI.

Overall, I had a lot of fun writing this project. The problem seems quite simple, but has lots of possible edge cases and challenges that have to be considered. I am proud of the work that I did and the finished project.